Hybrid High-order methods: Overview, implementation and latest developments

Matteo Cicuttin

École Nationale des Ponts et Chaussées (CERMICS) – Marne-la-Vallée
INRIA – Paris

CEA Saclay, 12/07/2019

Outline

1. Introduction to HHO
2. HHO in software: the DiSk++ library
3. HHO for advanced applications
    - The Unfitted HHO method
    - The Multiscale HHO method

Context

HHO belongs to the family of **Di**scontinuous **Sk**eletal methods.

Solution of BVPs is approximated by

- attaching unknowns to mesh faces $\Longrightarrow$ “skeletal”
- using polynomials discontinuous in the mesh skeleton $\Longrightarrow$ “discontinuous”

HHO uses also cells unknowns

- eliminated by local Shur complement

Context, schemes related to HHO

Low order:

- Non-conforming FEM [Crouzeix, Raviart '73]
- Mimetic finite differences [Brezzi, Lipnikov, Shashkov '05]
- Hybrid finite volumes [Droniou, Eymard, Gallouet, Herbin '06-'10]

High order:

- Hybridizable DG (HDG) [Cockburn, Gopalakrishnan, Lazarov '09]
- Non-conforming VEM [Lipnikov, Manzini '14]

HHO, HDG and ncVEM are closely related [Cockburn, Di Pietro, Ern, 16]

HHO features

- General mesh support
  - polygonal/polyhedral cells
  - hanging nodes
- Computational efficiency
  - HHO system size (3D) is $k^2\#(\text{faces})$, dG is $k^3\#(\text{cells})$
  - Compact stencil
  - $O(h^{k+1})$ energy error convergence
- Implementation friendly
  - Construction independent of spatial dimension and cell shape
  - Efficient implementation with generic programming
- Physical fidelity
  - HHO is locally conservative

Setting: Poisson model problem

Let $\Omega \subset \mathbb{R}^d$ with $d \in \{1, 2, 3\}$ be an open, bounded and connected polytopal domain. We will consider the model problem

$$\begin{aligned} -\Delta u &= f \qquad \text{in } \Omega, \\ u &= 0 \qquad \text{on } \partial\Omega, \end{aligned}$$

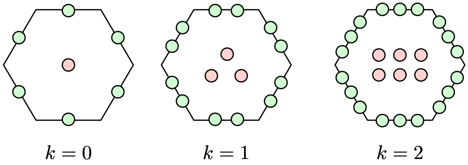
with $f \in L^2(\Omega)$. By setting $V := H_0^1(\Omega)$, its weak form is

$$\text{Find } u \in V \text{ such that } (\nabla u, \nabla v)_\Omega = (f, v)_\Omega \text{ for all } v \in V.$$

Other boundary conditions can be considered as well.

HHO Ingredient 0: Degrees of freedom

To discretize our problem we need a mesh. Then we choose the unknowns:



Unknowns: Polynomials of degree k attached to the cells and to the faces.

Let $\mathcal{M} := (\mathcal{T}, \mathcal{F})$ be the mesh consisting of the set of cells $\mathcal{T}$ and the set of faces $\mathcal{F}$ in which Ω is discretized. For each $T \in \mathcal{T}$ we can define the local space of DoFs

$$U_T^k := \mathbb{P}_d^k(T) \times \left\{ \bigtimes_{F \in \mathcal{F}_T} \mathbb{P}_{d-1}^k(F) \right\}$$

HHO ingredient 1: Reconstruction operator

High-order reconstruction

$$R_T^{k+1} : \underbrace{U_T^k}_{\text{cell/face dofs}} \rightarrow \underbrace{\mathbb{P}_d^{k+1}(T)}_{\text{higher-order poly}}$$

R_T^{k+1} solves for all $(v_T, v_{\partial T}) \in U_T^k$ and for all $w \in \mathbb{P}_d^{k+1}(T)$:

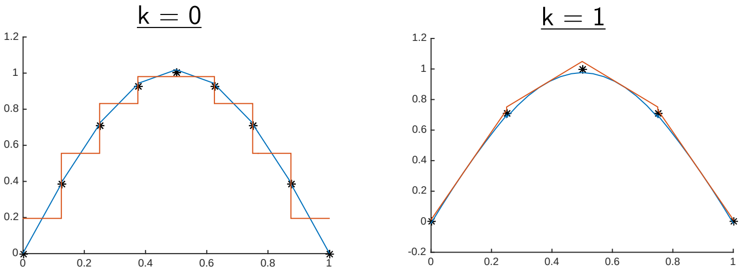
$$\begin{aligned}(\nabla R_T^{k+1}(v_T, v_{\partial T}), \nabla w)_T :&= -(v_T, \Delta w)_T + (v_{\partial T}, \boldsymbol{n}_T \cdot \nabla w)_F \\ &= (\nabla v_T, \nabla w)_T + (v_{\partial T} - v_T, \boldsymbol{n}_T \cdot \nabla w)_F\end{aligned}$$

together with the mean value condition $(R_T^{k+1}(v_T, v_{\partial T}), 1)_T = (v_T, 1)_T$.

$R_T^{k+1}(v_T, v_{\partial T})$ is computed by solving local Neumann problem.

The reconstruction in action

Consider the function $sin(\pi x)$ in $[0,1]$. We project it on U_T^k and then reconstruct, obtaining something in $\mathbb{P}_d^{k+1}(T)$:



Reconstruction is used to build bilinear form on $U_T^k \times U_T^k$:

$$a_T^{(1)}((v_T, v_{\partial T}), (w_T, w_{\partial T})) = (\nabla R_T^{k+1}(v_T, v_{\partial T}), \nabla R_T^{k+1}(w_T, w_{\partial T}))_T,$$

which mimics *locally* the l.h.s. of our original problem.

HHO ingredient 2: Stabilization operator

Stabilization needed: $\{\nabla \mathrm{R}_T^{k+1}(v_T, v_{\partial T}) = \mathbf{0}\} \not\Rightarrow \{v_T = v_{\partial T} = \text{const}\}$.

Setting $r_T^{k+1} := R_T^{k+1}(v_T, v_{\partial T})$, we introduce a penality on the difference between functions on faces and traces of functions in cell:

$$S_T^k(v_T, v_{\partial T}) := \Pi_{\partial T}^k \left((v_{\partial T} - v_T) + (I - \Pi_T^k) r_T^{k+1}(0, v_{\partial T} - v_T) \right).$$

This stabilization is a key feature of HHO: it gives h^{k+1} convergence in energy norm and h^{k+2} in L_2 norm (assuming elliptic regularity).

We introduce a second bilinear form on $U_T^k \times U_T^k$:

$$a_T^{(2)}((v_T, v_{\partial T}), (w_T, w_{\partial T})) = h_T^{-1}(S_T^k(v_T, v_{\partial T}), S_T^k(w_T, w_{\partial T}))_{\partial T},$$

where h_T denotes the diameter of the cell T.

Discrete problem

For all $T \in \mathcal{T}$, we combine reconstruction and stabilization bilinear forms into a_T on $U_T^k \times U_T^k$ such that

$$a_T := a_T^{(1)} + a_T^{(2)}.$$

We then do a standard cell-wise assembly

$$a_{\mathcal{M}}(u_{\mathcal{M}}, w_{\mathcal{M}}) := \sum_{T \in \mathcal{T}} a_T((u_T, u_{\partial T}), (w_T, w_{\partial T})),$$
$$\ell_{\mathcal{M}}(w_{\mathcal{M}}) := \sum_{T \in \mathcal{T}} (f, w_T)_T.$$

Finally we search for $u_{\mathcal{M}} := (u_{\mathcal{T}}, u_{\mathcal{F}}) \in U_{\mathcal{M},0}^k$ such that

$$a_{\mathcal{M}}(u_{\mathcal{M}}, w_{\mathcal{M}}) = \ell_{\mathcal{M}}(w_{\mathcal{M}}), \qquad \forall w_{\mathcal{M}} := (w_{\mathcal{T}}, w_{\mathcal{F}}) \in U_{\mathcal{M},0}^k,$$

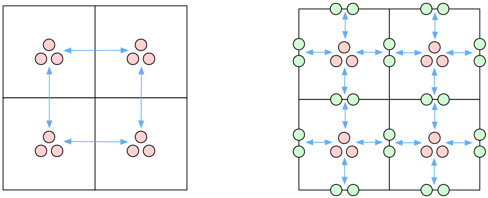
where Dirichlet BCs are imposed strongly on the face unknowns.

Cell-based unknowns are removed by local static condensation. Global problem has only face unknowns.

A remark on the stencil

We compare dG (left) and HHO (right). In HHO:

- Communication between cells mediated by face unknowns
- Assembly simpler, is more FEM-like than dG-like



HHO seems to use much more DoFs, but don't be fooled:

- Cell DoFs get statically condensed
- Face DoFs grow like $O(k|\mathcal{F}|)$ in 2D and $O(k^2|\mathcal{F}|)$ in 3D.

A remark on polynomial degree

In HHO we can use different polynomial degrees on cells and faces. Consider the space

$$U_T^{l,k} := \mathbb{P}_d^l(T) \times \left\{ \bigtimes_{F \in \mathcal{F}_T} \mathbb{P}_{d-1}^k(F) \right\}$$

where $k \geqslant 0$ is the degree of the face unknowns and $l \geqslant 0$ the one of the cell unknowns:

- $l = k$: standard, equal order case
- $l = k - 1$: same properties of equal order case $(k \geqslant 1)$
- $l = k + 1$:
  - again, same properties
  - Simpler stabilization, just $S_T^k(v_T, v_{\partial T}) := \Pi_{\partial T}^k (v_{\partial T} - v_T)$
  - More unknowns to statically condense

Thanks to this, p-refinement becomes easy.

Implementing DiSk methods, goals

As seen, HHO is formulated in a way that is

- Dimension-independent: we deal only with concepts of cells and faces, they have meaning in 1D, 2D, 3D
- Cell-shape-independent: we didn't make any assumption on cell shape

Mathematically this kind of formulation is natural.

Implementing DiSk methods, goals

As seen, HHO is formulated in a way that is

- Dimension-independent: we deal only with concepts of cells and faces, they have meaning in 1D, 2D, 3D
- Cell-shape-independent: we didn't make any assumption on cell shape

Mathematically this kind of formulation is natural.

To support HHO development, we wanted a software platform with the same level of generality. We created `DiSk++`, a platform that:

- fully supports HHO and is able to run it on any kind of mesh
- is efficient, on any kind of mesh
- allows the user to write its code without caring about the details of the underlying mesh (element shape/space dimension)

In one sentence: write the method once, run it on any kind of mesh - even the ones not supported yet.

`DiSk++` is open-source: `https://github.com/wareHHOuse/diskpp`

DiSk++

By using generic programming[1] `DiSk++` gives to the user a simple interface to code efficiently numerical methods like HHO, HDG, dG, ...

Generic programming is a technique that allows to write algorithms and data structures where also types are parameters. It enables to build zero cost abstractions.

Generic programming is about

- Code reuse: with the correct abstractions in place, the same code can be reused many times
- Performance: templated C++ code can frequently be optimized much more than C or Fortran
- Correctness: DiSk++ leverages the C++ type system to protect the user from vast classes of bugs
- ...

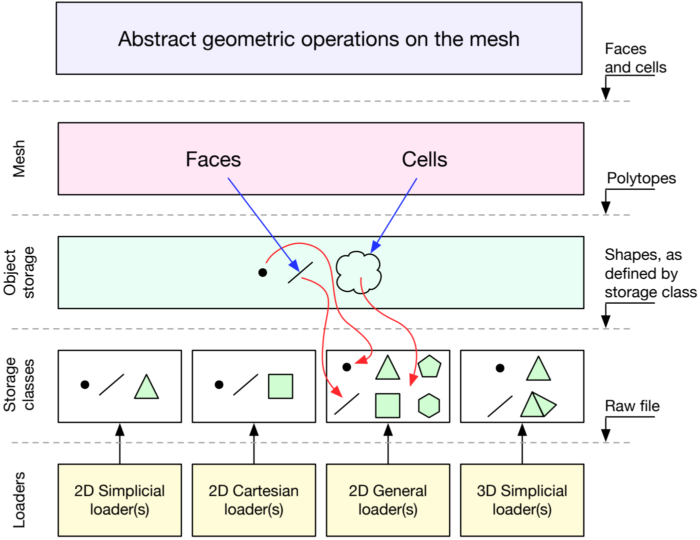
[1] MC, D. A. Di Pietro, A. Ern Implementation of Discontinuous Skeletal methods on arbitrary-dimensional, polytopal meshes using generic programming, J. Comp. Appl. Math. Vol. 334, 2018.

Abstractions in DiSk++

`DiSk++` is essentially a collection of abstraction layers to give an uniform set of operations on any mesh:

1. Mesh loading (from different file formats)
2. Mesh representation [biggest issue]
3. Geometric operations
4. Quadratures/Basis functions
5. Numerical methods components (i.e. HHO operators)

The overall architecture of DiSk++



Mesh element queries in DiSk++

Example: suppose we want to compute geometric properties of mesh elements.
DiSk++ allows code as general as:

```
for (auto& cl : msh) {
  //measure: volume, area, lenght depending on dimension
  auto cell_meas = measure(msh, cl);
  auto cell_bar = barycenter(msh, cl);
  auto fcs = faces(msh, cl); //get faces
  for (auto& fc : fcs) { //loop on them
    auto face_meas = measure(msh, fc);
    auto face_bar = barycenter(msh, fc);
  }
}
```

This code will work on any mesh you will throw at it as efficiently as possible! The abstraction is zero cost.

Quadratures and basis functions in DiSk++

Also quadratures and basis functions are generic:

```
for (auto& cl : msh) {
  auto basis = make_scalar_monomial_basis(msh, cl, degree);
  auto qps = integrate(msh, cl, 2 * degree);
  for (auto& qp : qps) {
    auto dphi = basis.eval_gradients(qp.point());
    stiffness_matrix += qp.weight() * dphi * trans(dphi);
  }
}
```

- Simplicial mesh $\Longrightarrow$ simplicial quadratures
- Cartesian mesh $\Longrightarrow$ tensorized Gauss points
- General mesh $\Longrightarrow$ split in simplices

Again, the user does not need to know anything about the underlying mesh.
You'll get automatically the right thing you need.

HHO, a layer on DiSk++

`DiSk++` is mostly "infrastructure" not tied to HHO.

HHO is just a thin layer over `DiSk++`: you can easily implement other methods.

Thanks to this infrastructure, HHO operators we discussed are implemented in a completely mesh- and dimension-independent fashon.

- Gradient reconstruction operator ($\approx$ 80 LOCs)
- Stabilization operator ($\approx$ 70 LOCs)

Debug and maintenance become much easier!

To solve a problem with HHO, just combine the blocks provided by the library!

Assembly of our diffusion problem

Going back to our initial toy problem, its assembly loop in DiSk++ reduces to

```
auto assembler = make_diffusion_assembler(msh, hdi);

for (auto& cl : msh) {
  auto cb   = make_scalar_monomial_basis(msh, cl, hdi);
  auto gr   = make_hho_scalar_laplacian(msh, cl, hdi);
  auto stab = make_hho_scalar_stabilization(msh, cl, gr, hdi);
  auto rhs  = make_rhs(msh, cl, cb, rhs_fun);
  auto A    = gr + stab;
  assembler.assemble(msh, cl, A, rhs, dirichlet_bc);
}

assembler.finalize();
```

Latest developments of DiSk++

Started in 2016, in the last year DiSk++ evolved a lot.

- It is a community effort. **Four main developers:** K. Cascavita, MC, G. Delay, N. Pignet. Join us on the wareHHOuse organization in GitHub:

  `https://github.com/wareHHOuse`

- Simplified some parts of the code, in particular the construction of HHO operators, much cleaner API now
- Added many automatic tests. We are able to detect automatically many problems/regressions in the code.
- Benchmarked some parts against other HHO implementations
- Speed improvements

Stay tuned: soon a guided HHO tutorial will be available!

Current capabilities of DiSk++

We added many new modules for

- different computational mechanics problems
- variants of the HHO method

DiSk++ allowed us to deploy HHO on many domains:

- Scalar diffusion (MC)
- Unfitted HHO (MC, GD)
- Linear elasticity (MC, NP)
- Eigenvalue problems (MC)
- Hyperelasticity (NP)
- Plasticity (NP)
- Bingham flows (KC)
- Signorini problem (KC)
- Obstacle problems (MC)
- Multiscale HHO (MC)

Unfitted HHO: model problem

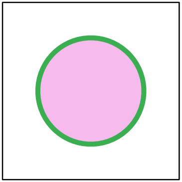
Let $\Omega \subset \mathbb{R}^d$ such that:

$$\bar{\Omega} = \bar{\Omega}^1 \cup \bar{\Omega}^2, \qquad \Gamma = \partial\Omega^1 \cap \partial\Omega^2$$

The following interface problem is considered:

$$\begin{cases} -\mathrm{div}(\kappa \nabla u) = f & \text{in } \Omega^1 \cup \Omega^2, \\ [\![u]\!]_\Gamma = g_D & \text{on } \Gamma, \\ [\![\kappa \nabla u]\!]_\Gamma \cdot \boldsymbol{n}_\Gamma = g_N & \text{on } \Gamma \end{cases}$$

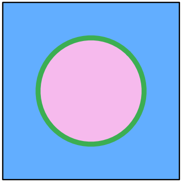
Fictitious domain problem



Diffusivity:

κ_1 in Ω^1

Interface problem



Diffusivity:

κ_1 in Ω^1

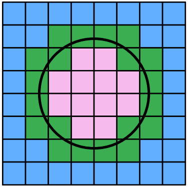
κ_2 in Ω^2

Unfitted meshing

Interface Γ described by a level set function or other techniques.

We want to mesh Ω without respecting Γ:

- Uncut cells of Ω^1
- Uncut cells of Ω^2 (if interface problem)
- Cells cut by Γ

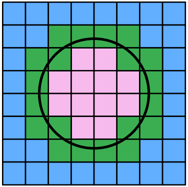


Unfitted meshing

Interface Γ described by a level set function or other techniques.

We want to mesh Ω without respecting Γ:

- Uncut cells of Ω^1
- Uncut cells of Ω^2 (if interface problem)
- Cells cut by Γ



Let T be a cut cell. Define $T^i = T \cap \Omega^i, i \in \{1, 2\}$

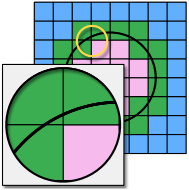
- High-contrast if $\min(\kappa_1, \kappa_2) \ll \max(\kappa_1, \kappa_2)$

Unfitted meshing

Interface Γ described by a level set function or other techniques.

We want to mesh Ω without respecting Γ:

- Uncut cells of Ω^1
- Uncut cells of Ω^2 (if interface problem)
- Cells cut by Γ



Let T be a cut cell. Define $T^i = T \cap \Omega^i, i \in \{1,2\}$

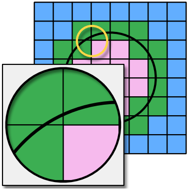
- High-contrast if $\min(\kappa_1, \kappa_2) \ll \max(\kappa_1, \kappa_2)$
- Degenerate cut if $\min(|T^1|, |T^2|) \ll \max(|T^1|, |T^2|)$

Unfitted meshing

Interface Γ described by a level set function or other techniques.

We want to mesh Ω without respecting Γ:

- Uncut cells of Ω^1
- Uncut cells of Ω^2 (if interface problem)
- Cells cut by Γ



Let T be a cut cell. Define $T^i = T \cap \Omega^i, i \in \{1, 2\}$

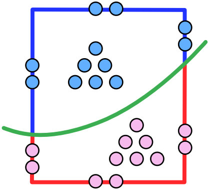
- High-contrast if $\min(\kappa_1, \kappa_2) \ll \max(\kappa_1, \kappa_2)$
- Degenerate cut if $\min(|T^1|, |T^2|) \ll \max(|T^1|, |T^2|)$

HHO provides robustness

- w.r.t. high contrast, via diffusion-dependent averaging [Ern, Stephansen, Zunino '09]
- w.r.t. degenerate cuts, via agglomeration → very natural for HHO.

Unfitted HHO unknowns

- Uncut cells: standard HHO unknowns
- Cut cells: a pair of HHO unknowns → one function on each side of the cut
- No unknowns on the cut



In detail, on cut cells unknowns are

$$\hat{V}_T = (V_T, \partial V_T) = ((v_{T^1}, v_{T^2}), (v_{\partial T^1}, v_{\partial T^2})) \in \hat{\mathcal{X}}_T$$

belonging to the space

$$\hat{\mathcal{X}}_T = \Big(\big(\mathbb{P}^{k+1}(T^1) \times \mathbb{P}^{k+1}(T^2)\big) \times \big(\mathbb{P}^{k}(\mathcal{F}_{(\partial T)^1}) \times \mathbb{P}^{k}(\mathcal{F}_{(\partial T)^2})\big) \Big).$$

but how do we join the two sides?

Nitsche mortaring

To "glue together" the two sides we use standard Nitsche mortaring:

$$n_T(V,W) = \sum_{i\in\{1,2\}} \int_{T^i} \kappa^i \nabla v^i \cdot \nabla w^i + \int_{T^\Gamma} \eta \frac{\kappa^1}{h_T} [\![V]\!]_\Gamma [\![W]\!]_\Gamma$$
$$- \int_{T^\Gamma} (\kappa \nabla v)^1 \cdot \boldsymbol{n}_\Gamma [\![W]\!]_\Gamma + (\kappa \nabla w)^1 \cdot \boldsymbol{n}_\Gamma [\![V]\!]_\Gamma$$

Penalization η has to be taken large enough, as in dG.

Unfitted HHO operators

As usual, reconstruction maps from HHO unknowns to polynomials

$$R_T^{k+1} \;:\; \underbrace{\hat{\mathcal{X}}}_{\text{cutHHO unknown}} \longrightarrow \underbrace{\mathbb{P}^{k+1}(T^1) \times \mathbb{P}^{k+1}(T^2)}_{\text{one poly per cut side}}$$

Let $\hat{V}_T = (V_T, V_{\partial T}) \in \hat{\mathcal{X}}$. The reconstruction is obtained by solving, for all $Z \in \mathbb{P}^{k+1}(T^1) \times \mathbb{P}^{k+1}(T^2)$:

$$n_T(R_T^{k+1}(\hat{V}_T), Z) = n_T(V_T, Z) - \sum_{i\in\{1,2\}} \int_{(\partial T)^i} (v_{T^i} - v_{(\partial T)^i})\boldsymbol{n} \cdot \kappa^i \nabla z^i$$

The stabilization is done as usual by penalizing difference between trace of function on cells and function on faces:

$$s_T(\hat{V}_T, \hat{W}_T) := \sum_{i\in\{1,2\}} \kappa^i h_T^{-1} \int_{(\partial T)^i} \Pi^k_{(\partial T)^i}\left((v_{T^i} - v_{(\partial T)^i})\right)\left(w_{T^i} - w_{(\partial T)^i}\right)\big) .$$

Discrete problem

Two cases to handle in discrete problem assembly.

- Cut cells $T \in \mathcal{T}^{\Gamma}$:

$$\hat{a}_T^{\Gamma}(\hat{V}_T, \hat{W}_T) = n_T(R_T^{k+1}(\hat{V}_T), R_T^{k+1}(\hat{W}_T)) + s_T(\hat{V}_T, \hat{W}_T)$$
$$\hat{l}_T^{\Gamma}(\hat{W}_T) = \sum_{i\in\{1,2\}} \int_{T^i} f w_{T_i} + \int_{T^{\Gamma}} g_N w_{T^2} + g_D \Phi_T(W_T)$$

where $\Phi_T(W_T) = -\kappa^1 \nabla w_{T^1} \cdot \boldsymbol{n}_{\Gamma} + \eta \kappa^1 h_T^{-1} [\![W_T]\!]_{\Gamma}$ (cf. dG)

- Uncut cells $T \in \mathcal{T}^{\backslash\Gamma}$:

$$\hat{a}_T^{\backslash\Gamma}(\hat{v}_T, \hat{w}_T) = a_T(r_T^{k+1}(\hat{v}_T), r_T^{k+1}(\hat{w}_T)) + s_T(\hat{v}_T, \hat{w}_T)$$
$$\hat{l}_T^{\backslash\Gamma}(\hat{w}_T) = \int_T f w_T$$

Cell unknowns are statically condensed as in regular HHO.

Discrete problem, assembly

The discrete problem is assembled pretty much like standard HHO.
Global space of unknowns of Ω^i:

$$\hat{\mathcal{X}}_h^i = \mathbb{P}^{k+1}(\mathcal{T}^i) \times \mathbb{P}^k(\mathcal{F}^i), \qquad i \in \{1, 2\}$$

Global space on Ω is $\hat{\mathcal{X}}_h = \hat{\mathcal{X}}_h^1 \cup \hat{\mathcal{X}}_h^2$. We can consider $\hat{\mathcal{X}}_{h0}$ where we enforce Dirichlet on face unknowns.
Global bilinear forms are:

$$\hat{a}_h(\hat{V}_h, \hat{W}_h) = \sum_{T \in \mathcal{T}^{\setminus\Gamma}} \hat{a}_T^{\setminus\Gamma}(\hat{v}_T, \hat{w}_T) + \sum_{T \in \mathcal{T}^{\Gamma}} \hat{a}_T^{\Gamma}(\hat{V}_T, \hat{W}_T)$$

$$\hat{l}_h(\hat{W}_h) = \sum_{T \in \mathcal{T}^{\setminus\Gamma}} \hat{l}_T^{\setminus\Gamma}(\hat{w}_T) + \sum_{T \in \mathcal{T}^{\Gamma}} \hat{l}_T^{\Gamma}(\hat{W}_T)$$

We look for $\hat{V}_h \in \hat{\mathcal{X}}_{h0}$ s.t. $\hat{a}_h(\hat{V}_h, \hat{W}_h) = \hat{l}_h(\hat{W}_h), \forall \hat{W}_h \in \hat{\mathcal{X}}_{h0}$.

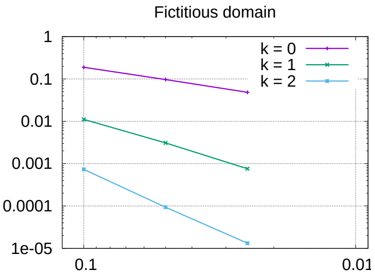
Numerical results

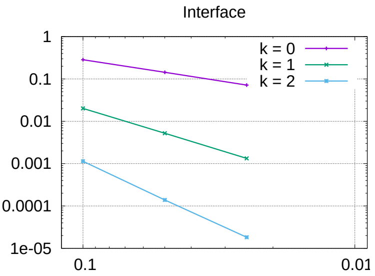
If $U^{ex} = (u^1, u^2) \in H^{k+2}(\Omega^1) \times H^{k+2}(\Omega^2)$, error estimate is

$$\sum_{T \in \mathcal{T}^{\setminus \Gamma}} \kappa_T ||\nabla(u^{ex} - u_T)||_T^2 + \sum_{T \in \mathcal{T}^{\Gamma}} \sum_{i \in \{1,2\}} \kappa^i ||\nabla(U^{ex} - U_T)^i||_T^2$$

$$+ \sum_{T \in \mathcal{T}^{\Gamma}} \frac{\kappa^1}{h_T} ||g_D - [\![U_\Gamma^{ex}]\!]||_{T^\Gamma}^2 + \frac{h_T}{\kappa^2} ||g_N - [\![\kappa \nabla U_\Gamma^{ex}]\!] \cdot \boldsymbol{n}_\Gamma||_{T^\Gamma}^2$$

$$\leqslant \sum_{i \in \{1,2\}} \kappa^i h^{2(k+1)} |u^i|_{H^{k+2}(\Omega^i)}^2$$





The Multiscale HHO method

Let $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$; $\varepsilon > 0$ and much smaller than the length scale ℓ_Ω of Ω. We consider

$$\begin{cases} -\mathrm{div}(\mathbb{A}_\varepsilon \nabla u_\varepsilon) = f & \text{in } \Omega, \\ u_\varepsilon = 0 & \text{on } \partial\Omega, \end{cases}$$

where $f \in L^2(\Omega)$ is non-oscillatory and $\mathbb{A}_\varepsilon$ is an oscillatory, uniformly elliptic and bounded matrix-valued field on Ω.

- Monoscale methods: too many DoFs needed to resolve $\mathbb{A}_\varepsilon$
- Multiscale methods come to rescue: encode the oscillations of $\mathbb{A}_\varepsilon$ in the basis functions of the approximation space, and approximate u_ε on a coarse mesh $\mathcal{T}_H$ with $\varepsilon \leqslant H \leqslant \ell_\Omega$

msHHO

Ingredients of the msHHO method:

- Discrete unknowns are polynomials of order $k \geqslant 0$ on faces and $l \geqslant 0$ on cells of $\mathcal{T}_H$ (as in monoscale HHO)
- Oscillatory basis functions encoding $\mathbb{A}_\varepsilon$ for cells and faces
- Reconstruction operator based on oscillatory basis functions

Two variants of the method: equal order method $(l = k)$ and mixed order method $(l = k - 1)$

Literature

- prior art for $k = 0$: msFEM à la Crouzeix–Raviart [Le Bris, Legoll, Lozinski 13]
- msHHO method provides an extension: arbitrary order and polytopal meshes

msHHO ingredient 1: Oscillatory basis functions

msHHO approximation space is

$$V_\varepsilon^{k+1}(T) = \{v \in H^1(T) \mid \nabla\cdot(\mathbb{A}_\varepsilon \nabla v) \in \mathbb{P}^{k-1}(T),\ \boldsymbol{n}_T\cdot\mathbb{A}_\varepsilon\nabla v \in \mathbb{P}^k(\partial T)\}$$

The basis functions of $V_\varepsilon^{k+1}(T)$ are

- Cell basis functions (for $k \geqslant 1$)

$$\varphi_{\varepsilon,T}^{k+1,i} = \underset{\substack{\varphi\in H^1(T) \\ \Pi_F^k(\varphi)=0,\ \forall F\subset\partial T}}{\arg\min} \int_T \left[\frac{1}{2}\mathbb{A}_\varepsilon\nabla\varphi\cdot\nabla\varphi - \Phi_T^{k-1,i}\varphi\right]$$

  where $(\Phi_T^{k-1,i})_{1\leqslant i\leqslant N_d^{k-1}}$ is a basis of $\mathbb{P}^{k-1}(T)$

- and Face basis functions (for $k \geqslant 0$)

$$\varphi_{\varepsilon,T,F}^{k+1,j} = \underset{\substack{\varphi\in H^1(T) \\ \Pi_F^k(\varphi)=\Phi_F^{k,j} \\ \Pi_\sigma^k(\varphi)=0,\ \forall\sigma\subset\partial T\setminus\{F\}}}{\arg\min} \int_T \left[\frac{1}{2}\mathbb{A}_\varepsilon\nabla\varphi\cdot\nabla\varphi\right]$$

  where $(\Phi_F^{k,j})_{1\leqslant j\leqslant N_{d-1}^k}$ is a basis of $\mathbb{P}^k(F)$

msHHO ingredient 2: Multiscale reconstruction operator

Given a pair $(v_T, v_{\partial T}) \in \mathbb{P}^l(T) \times \mathbb{P}^k(\partial T)$, the reconstruction returns an object of $V_\varepsilon^{k+1}(T)$.

$$R_{\varepsilon,T}^{k+1} \; : \; \underbrace{\mathbb{P}^l(T) \times \mathbb{P}^k(\partial T)}_{\text{cell and face unknowns}} \longrightarrow \underbrace{V_\varepsilon^{k+1}(T)}_{\text{oscillatory function}}$$

- $r := R_{\varepsilon,T}^{k+1}(v_T, v_{\partial T}) \in V_\varepsilon^{k+1}(T)$ solves, $\forall w \in V_\varepsilon^{k+1}(T)$,

  $$(\mathbb{A}_\varepsilon \nabla r, \nabla w)_{\boldsymbol{L}^2(T)} = -(v_T, \nabla \cdot (\mathbb{A}_\varepsilon \nabla w))_{L^2(T)} + (v_{\partial T}, \boldsymbol{n}_T \cdot \mathbb{A}_\varepsilon \nabla w)_{L^2(\partial T)}$$

  together with the mean-value condition $(r, 1)_{L^2(T)} = (v_T, 1)_{L^2(T)}$

- Oscillatory basis functions and $R_{\varepsilon,T}^{k+1}$ precomputed offline by meshing T (of size $H > \varepsilon$) with subcells of size $h < \varepsilon$, and using a mono-scale method to approximate the minimizers
- Also in msHHO, reconstruction operator used to mimic problem l.h.s.

Mixed-order msHHO

Consider the mixed-order variant of msHHO, where $l = k - 1$

- The local msHHO bilinear form is

$$\hat{a}_{\varepsilon,T}(\cdot,\cdot) = (\mathbb{A}_\varepsilon \nabla R^{k+1}_{\varepsilon,T}(\cdot), \nabla R^{k+1}_{\varepsilon,T}(\cdot))_{\boldsymbol{L}^2(T)}$$

- Mixed order method requires no stabilization: we explored the whole space $V^{k+1}_\varepsilon(T)$ to resolve the oscillatory nature of the problem

- Error estimate:

$$\left(\sum_{T \in \mathcal{T}_H} \|\mathbb{A}_\varepsilon^{\frac{1}{2}} \nabla(u_\varepsilon - R^{k+1}_{\varepsilon,T}(u_T, u_{\partial T}))\|^2_{\boldsymbol{L}^2(T)} \right)^{\frac{1}{2}} \leqslant c\left(\varepsilon^{\frac{1}{2}} + H^{k+1} + (\varepsilon/H)^{\frac{1}{2}}\right)$$

where the right-hand side term is the usual resonance error.

Equal-order msHHO

Consider the equal-order variant of msHHO, where $l = k$

- We still reconstruct in $V_\varepsilon^{k+1}(T)$ using $R_{\varepsilon,T}^{k+1}$
- The local msHHO bilinear form is

$$\hat{a}_{\varepsilon,T}(\cdot,\cdot) = (\mathbb{A}_\varepsilon \nabla R_{\varepsilon,T}^{k+1}(\cdot), \nabla R_{\varepsilon,T}^{k+1}(\cdot))_{\boldsymbol{L}^2(T)} + h_T^{-1}(S_{\varepsilon,\partial T}^k(\cdot), S_{\varepsilon,\partial T}^k(\cdot))_{L^2(\partial T)}$$

$$S_{\varepsilon,\partial T}^k(v_T, v_{\partial T}) = v_T - \Pi_T^k(R_{\varepsilon,T}^{k+1}(v_T, v_{\partial T}))$$

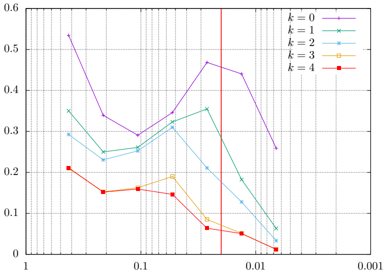
  - stabilization needed because we reconstruct in $V_{\varepsilon,T}^{k+1}$ and not in $\tilde{V}_{\varepsilon,T}^{k+1} = \{v \in H^1(T) \mid \nabla\cdot(\mathbb{A}_\varepsilon \nabla v) \in \mathbb{P}^k(T),\ \boldsymbol{n}_T\cdot\mathbb{A}_\varepsilon \nabla v \in \mathbb{P}^k(\partial T)\}$
  - stabilization can be avoided by computing additional oscillatory basis functions to span $\tilde{V}_{\varepsilon,T}^{k+1}$; see [Le Bris, Legoll, Lozinski 14] for $k = 0$ (one additional basis function)

- Error estimate: same as in mixed-order case

Numerical experiment

- Periodic setting with $\mathbb{A}_\varepsilon(x,y) = a(x/\varepsilon, y/\varepsilon)\mathbb{I}_2$, $\varepsilon = \pi/150 \approx 0.02$, $a(x,y) = 1 + 100\cos^2(\pi x)\sin^2(\pi y)$
- Hierarchical triangular meshes of size $H_l = 0.43 \times 2^{-l}$, $l \in \{0{:}9\}$
  - resonance expected for $H_4 > \varepsilon > H_5$
  - reference solution computed for $l_{\text{ref}} = 9$ and $k_{\text{ref}} = 2$
  - cell problems: mono-scale HHO, degree 1, mesh level $l = 8$

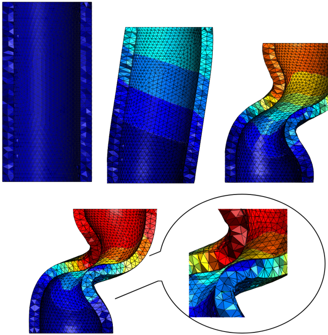
Energy error (relative) as a function of H_l, equal-order msHHO, $k \in \{0, \dots, 4\}$



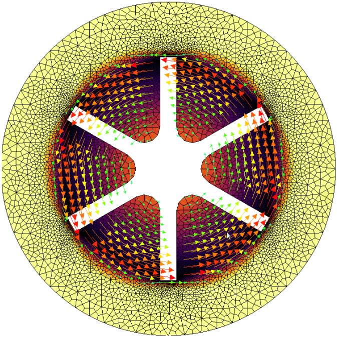
Conclusions

- HHO has different assets to offer:
  - competitive computational cost
  - one formulation supports completely general meshes in any dimension
  - physical fidelity
  - implementation-friendly
- Widely deployed on many classes of problems
- Software library available: https://github.com/wareHHOuse/

Thank you for your attention!



N. Pignet, Large deformations of a sheared cylinder



K. Cascavita, Bingham fluid in a vane cavity

e-mail: `matteo.cicuttin@enpc.fr`

code: `https://github.com/wareHHOuse/diskpp`